

2000 MICROVAX VER 1.000

ALIST A
NAME YFF
WOST

THIS FILE WAS MODIFIED BY MIKE LORENZEN TO LEAVE ONLY
THE SOURCE CODE THAT PRODUCES THE 2K ROM MATH PACKAGE.
IN THE ATARI OPERATING SYSTEM.

SEPT 6, 1981

TITLE 'SHEPARDSON CONTRACTED MATH PACKAGE'
2K ROM MATH PACKAGE

(C) COPYRIGHT 1978 SHEPARDSON MICROSYSTEMS, INC.

FPORG EQU \$D800
FPREC EQU 6
FMPREC EQU FPREC-1

LENGTH OF FLOATING POINT MANTISA

```

1
2
3      .ALIST A
4      .NAME 'FP'
5      .ASET
6
7      THIS FILE WAS MODIFIED BY MIKE LORENZEN TO LEAVE ONLY
8      THE SOURCE CODE THAT PRODUCES THE 2K ROM MATH PACKAGE BDC
9      IN THE ATARI OPERATING SYSTEM
10
11      SEPT  6, 1981
12
13      TITLE 'SHEPARDSON CONTRACTED MATH PACKAGE '
14
15      2K ROM MATH PACKAGE
16
17      *****
18      (C) COPYRIGHT 1978 SHEPARDSON MICROSYSTEMS, INC
19      *****
20
21      .DS00      EQU      *DB00
22      .D000      EQU      0
23      .D001      EQU      *DB00-1      : LENGTH OF FLOATING POINT MANISSA

```

```

21 /
22 /
23 /
24 /
25 /
26 0004 FR0 RES 1 / FP REG0
27 0005 FROM RES FPREG-1 / FP REG0 MANT
28 000A FRE RES FPREG / FP REG0 EXP
29 00E0 FR1 RES 1 / FP REG 1
30 00E1 FR1M RES FPREG-1 / FP REG1 MANT
31 00E6 FR2 RES FPREG / FP REG 2
32 00EC FRX RES 1 / FP SPARE
33 /
34 /
35 /
36 00ED EEXP RES 1 / VALUE OF E
37 00EE FRSIGN EQU * / FP SIGN
38 00EE NSIGN RES 1 / SIGN OF #
39 00EF SGRcnt PLYCNT
40 00EF PLYCNT
41 00EF ESIGN RES 1 / SIGN OF EXPONENT
42 00F0 SGNFLG
43 00F0 FCHRFLG RES 1 / 1ST CHAR FLAG
44 00F1 XFMRFLG
45 00F1 DIGRT RES 1 / # OF DIGITS RIGHT OF DECIMAL
46 /
47 /
48 /
49 00F2 CIX RES 1 / CURRENT INPUT INDEX
50 00F3 INBUFF RES 2 / LINE INPUT BUFFER
51 /
52 /
53 /
54 00F5 ZTEMP1 RES 2 / LOW LEVEL ZERO PAGE TEMPS
55 00F7 ZTEMP4 RES 2
56 00F9 ZTEMP3 RES 2
57 /
58 00FB DEGFLG
59 00FB RADFLG RES 1 / 0=RADIANS, 6= DEGREES
60 0000 RADON EQU 0 / INDICATE RADIANS
61 0006 DEGON EQU 6 / INDICATES DEGREES
62 00FC FLPTR RES 2
63 00FE FPTR2 RES 2

```

FLOATING POINT WORK AREA

RAM FOR ASCII TO FLOATING POINT CONVERSION

INPUT BUFFER

```
64 PAGE 'FLOATING POINT RAM (NON-ZERO PAGE)'  
65  
66 MISC NON-ZERO PAGE RAM  
67 USED FOR VALUES NOT ACCESSED FREQUENTLY  
68 ORG $57E  
69 057E LBPR1 RES 1 ; LBUFF PREFIX 1  
70 057F LBPR2 RES 1 ; BLUFF PREFIX 2  
71 0580 LBUFF RES 128 ; LINE BUFFER  
72  
73 ORG LBUFF+$60  
74 05E0 PLYARG RES FPREC  
75 05E6 FPSCR RES FPREC  
76 05EC FPSCR1 RES FPREC  
77 05E6 FSCR EQU FPSCR  
78 05EC FSCR1 EQU FPSCR1
```



```

79                                     PAGE 'FLOATING POINT'
80                                     ORG  FPDORG
81                                     LOCAL
82
83                                     ASCIN - CONVERT ASCII INPUT TO INTERNAL FORM
84
85                                     ON ENTRY  INBUFF - POINTS TO BUFFER WITH ASCII
86                                     CIX - INDEX TO 1ST BYTE OF #
87
88                                     ON EXIT   CC SET - CARRY SET IF NOT #
89                                     CARRY CLEAR OF #
90
91
92 D800                                AFP
93 D800                                CVAFP
94 D800                                ASCIN
95 D800 20 A1 DB                      JSR     SKPBLK
96 D803 20 BB DB                      JSR     _TSTCHAR ; SEE IF THIS COULD BE A NUMBER
97 D806 80 39                          BCS     _NONUM  ; BR IF NOT A NUMBER
98
99                                     SET INITIAL VALUES
100
101 D808 A2 ED                          LDX     #EEXP   ; ZERO 4 VALUES
102 D80A A0 04                          LDY     #4      ; X
103 D80C 20 48 DA                      JSR     ZXLY   ; X
104 D80F A2 FF                          LDX     #$FF
105 D811 86 F1                          STX     DIGRT  ; SET TO $FF
106
107 D813 20 44 DA                      JSR     ZFRO   ; CLEAR FRO
108
109 D816 F0 04                          BEQ     _IN2   ; UNCONDITIONAL BR
110
111
112 D818                                _IN1
113 D818 A9 FF                          LDA     #$FF   ; SET 1ST CHAR FLAG TO NON ZERO
114 D81A 85 F0                          STA     FCHRFLG ; X
115
116 D81C                                _IN2
117 D81C 20 94 DB                      JSR     _GETCHAR ; GET INPUT CHAR
118 D81F 80 21                          BCS     _NON1  ; BR IF CHAR NOT NUMBER
119
120
121                                     IT'S A NUMBER
122
123 D821 88                               PHA     ; SAVE ON CPU STACK
124 D822 A5 D5                          LDX     FROM   ; GET 1ST BYTE
125 D824 D0 11                          BNE     _INCE  ; INCR EXPONENT
126
127 D826 20 EB DB                      JSR     NIBSHO ; SHIFT FRO ONE NIBBLE LEFT
128
129 D82F 68                               PLA     ; GET DIGIT ON CPU STACK
130 D83A D5 D9                          ORA     FROM+FMPPREC-1 ; OR INTO LAST BYTE
131 D83C 85 D9                          STA     FROM+FMPPREC-1 ; SAVE AS LAST BYTE
132

```

ERR LINE ADDR B1 B2 B3 B4

FLOATING POINT

PAGE 5

```
133 ; COUNT CHARACTERS AFTER DECIMAL POINT
134 ;
135 D82E A6 F1 LDX DIGRT ; GET # OF DIGITS RIGHT
136 D830 30 E6 BMI _IN1 ; IF = $FF, NO DECIMAL POINT
137 D832 E8 INX ; ADD IN THIS CHAR
138 D833 86 F1 STX DIGRT ; SAVE
139 D835 D0 E1 BNE _IN1 ; GET NEXT CHAR
140 ;
141 ;
142 ; INCREMENT # OR DIGIT MORE THAN 9
143 ;
144 ;
145 D837 _INCE PLA ; CLEAR CPU STACK
146 D837 68 LDX DIGRT ; HAVE DP?
147 D838 A6 F1 BPL _INCE2 ; IF YES, DON'T INCR E COUNT
148 D83A 10 02 INC EEXP ; INCR EXPONENT
149 D83C E6 ED _INCE2 JMP _IN1 ; GET NEXT CHAR
150 D83E ;
151 D83E 4C 18 D8 ;
152 ;
153 ;
154 D841 _NONUM RTS ; RETURN FAIL
155 D841 60
```

```

156
157
158
159
160 DB42
161 DB42 C9 2E
162 DB44 F0 14
163 DB46 C9 45
164 DB48 F0 19
165
166 DB4A A6 F0
167 DB4C D0 68
168 DB4E C9 28
169 DB50 F0 C6
170 DB52 C9 2D
171 DB54 F0 00
172
173 DB56
174 DB56 B5 EE
175 DB58 F0 BE
176
177 DB5A
178 DB5A A6 F1
179 DB5C 10 58
180 DB5E E8
181 DB5F B6 F1
182 DB61 F0 B5
183
184 DB63
185 DB63 A5 F2
186 DB65 B5 EC
187 DB67 20 94 DB
188 DB6A B0 37
189
190
191
192 DB6C
193 DB6C AA
194 DB6D A5 ED
195 DB6F 48
196 DB70 B6 ED
197 DB72 20 94 DB
198
199
200 DB75 B0 17
201 DB77 48
202
203 DB78 A5 ED
204 DB7A 0A
205 DB7B B5 ED
206 DB7D 0A
207 DB7E 0A
208 DB7F B5 ED
209 DB81 B5 ED

```

PAGE

NON-NUMERIC IN NUMBER BODY

_NON1

```

CMP #1
BEG _DP
CMP #E
BEG _EXP
LDX FCHFLG
BNE _EXIT
CMP #'+'
BEG _IN1
CMP #'-'
BEG _MINUS

```

_MINUS

```

STA NSIGN
BEG _IN1

```

_DP

```

LDX DIGRT
BPL _EXIT
INX
STX DIGRT
BEG _IN1

```

_EXP

```

LDA CIX
STA FRX
JSR _GETCHAR
BCS _NON2

```

IT'S A NUMBER IN AN EXPONENT

_EXP2

```

TAX
LDA EEXP
PHA
STX EEXP
JSR _GETCHAR

```

_EXP3

```

BCS _EXP3
PHA
LDA EEXP
ASLA
STA EEXP
ASLA
ASLA
ADC EEXP
STA EEXP

```

ERR LINE ADDR B1 B2 B3 B4

FLOATING POINT

PAGE 7

210	D883	68			PLA		; GET SECOND DIGIT
211	D884	18			CLC		
212	D885	65 ED			ADC	EEXP	; GET EXPONENT INPUTTED
213	D887	85 ED			STA	EEXP	; SAVE
214							
215	D889	A4 F2			LDY	GIX	; INC TO NEXT CHAR
216	D88B	20 9D DB			JSR	_GCHR1	; X
217							
218							
219	D88E				__EXP3		
220	D89E	A5 EF			LDA	ESIGN	; GET SIGN OF EXPONENT
221	D890	F0 09			BEG	__EXP1	; IF NO SIGN, IT IS +
222	D892	A5 ED			LDA	EEXP	; GET EXPONENT ENTERED
223	D894	49 FF			EOR	#\$FF	; COMPLEMENT TO MAKE MINUS
224	D896	18			CLC		; X
225	D897	69 01			ADC	#1	; X
226	D899	85 ED			STA	EEXP	; SAVE
227	D89B				__EXP1		
228	D89B	68			PLA		; GET # DIGITS MORE THAN 9
229	D89C	18			CLC		; CLEAR CARRY
230	D89D	65 ED			ADC	EEXP	; ADD IN ENTERED EXPONENT
231	D89F	85 ED			STA	EEXP	; SAVE EXPONENT
232	D8A1	D0 13			BNE	_EXIT	; UNCONDITIONAL BR

233
 234
 235
 236
 237 DBA3
 238 DBA3 C9 28
 239 DBA5 F0 06
 240 DBA7 C9 2D
 241 DBA9 D0 07
 242
 243
 244 DBAB
 245 DBAB 85 EF
 246 DBAD
 247 DBAD 20 94 DB
 248 DBB0 90 BA
 249
 250
 251
 252
 253
 254 DBB2
 255 DBB2 A5 EC
 256 DBB4 85 F2
 257
 258

PAGE
 NON-NUMERIC IN EXPONENT
 NON2
 CMP #'+ , IS IT PLUS?
 BEQ _EPLUS , IF YES BR
 CMP #'-' , IS IT A MINUS?
 BNE _NOTE , IF NOT, BR
 _EMIN
 STA ESIGN , SAVE EXPONENT SIGN
 _EPLUS
 JSR _GETCHAR , GET CHARACTER
 BCC _EXP2 , IF A #, GO PROCESS EXPONENT
 E NOT PART OF OUR #
 _NOTE
 LDA FRX , POINT TO 1 PAST E
 STA CIX , RESTORE CIX
 FALL THRU TO EXIT

```

259                                     PAGE
260
261                                     WHOLE # HAS BEEN INPUTTED
262
263 DB06                                EXIT
264
265                                     BACK UP ONE CHAR
266
267 DB86 CA F2                          DEC    CIX          ; DECREMENT INDEX
268
269
270                                     CALCULATE POWER OF 10 = EXP - DIGITS RIGHT
271                                     WHERE EXP = ENTERED EXPONENT (COMPLEMENT OF -)
272                                     + # DIGITS MORE THAN 9
273
274 DB08 A5 ED                          LDA    EEXP          ; GET EXPONENT
275 DB8A A5 F1                          LDX    DIGRT        ; GET # DIGITS RIGHT OF DECIMAL
276 DB8C 30 05                          BMI    _EXIT1       ; NO DECIMAL POINT
277 DB8E F0 03                          BEQ    _EXIT1       ; # OF DIGITS AFTER D.P. = 0
278 DB80 38                             SEC          ; GET EXP - DIGITS RIGHT AS NEW EXP
279 DB01 E3 F1                          SBC    DIGRT        ; X
280
281                                     SHIFT RIGHT ALGEBRAIC TO DIVIDE BY 2 = POWER OF 100
282
283 DB03                                _EXIT1
284 DB03 48                             PHA
285 DB04 2A                             ROLA          ; SET CARRY WITH SIGN OF EXPONENT
286 DB05 68                             PLA          ; GET EXPONENT AGAIN
287 DB06 5A                             RORA          ; SHIFT RIGHT
288 DB07 85 ED                          STA    EEXP        ; SAVE POWER OF 100
289 DB09 90 03                          BCC    _EVEN       ; IF NO CARRY # EVEN
290
291 DB0B 20 EB DB                        JSR    NIBSHO       ; ELSE SHIFT 1 NIBBLE LEFT (MULT BY 10)
292 DB0E                                _EVEN
293 DB0E A5 ED                          LDA    EEXP        ; ADD 40 FOR EXCESS 64 + 4 FOR NORM
294 DB00 12                             CLC          ; X
295 DB01 69 44                          ADC    #$44        ; X
296 DB03 85 D4                          STA    FRO        ; SAVE AS EXPONENT
297
298 DB05 20 00 DC                        JSR    NORM        ; NORMALIZE NUMBER
299 DB08 80 0B                          BCS    _IND2       ; IF CARRY SET, IT'S AN ERROR
300
301                                     SET MANTISSA SIGN
302
303 DB0A A6 EE                          LDX    NSIGN       ; IS SIGN OF # MINUS?
304 DB0C F0 06                          BEQ    _INDON      ; IF NOT, BR
305
306 DB0E A5 D4                          LDA    FRO        ; GET EXPONENT
307 DB00 09 80                          ORA    #$80        ; TURN ON MINUS # BIT
308 DB02 85 D4                          STA    FRO        ; SET IN FRO EXP
309 DB04                                _INDON
310 DB04 18                             CLC          ; CLEAR CARRY
311 DB05                                _IND2
312 DB05 60                             RTS

```

```

313                                     PAGE
314
315 FPASC - CONVERT FLOATING POINT TO ASCII
316
317 ON ENTRY  FRO - # TO CONVERT
318
319 ON EXIT   INBUFF - POINTS TO START OF #
320           HIGH ORDER BIT OF LAST BYTE IS ON
321
322
323 DBE6 D8E6 CVPASC
324 DBE6 FASC
325 DBE6 20 31 DA JSR INTLBF , SET INBUFF TO PT TO LBUFF
326
327 DBE9 A9 30 LDA #'0' , GET ASCII ZERO
328 DBE8 8D 7F 05 STA LBUFF-1 , PUT IN FRONT OF LBUFF
329
330 TEST FOR E FORMAT REQUIRED
331
332 DBEE A5 D4 LDA FRO , GET EXPONENT
333 DBF0 F0 28 BEQ _EXPO , IF EXP = 0, # = 0, SO BR
334 DBF2 29 7F AND #$7F , AND OUT SIGN
335 DBF4 C9 3F CMP #$3F , IS IT LESS THAN 3F
336 DBF6 90 28 BCC _EFORM , IF YES, E FORMAT REQUIRED
337 DBF8 C9 45 CMP #$45 , IF IT IS > 44
338 DBFA B0 24 BCS _EFORM , IF YES, E FORMAT REQUIRED
339
340 PROCESS NOT E FORMAT
341
342 DBFC 35 SEC , SET CARRY
343 DBFD E9 3F SBC #$3F , GET DECIMAL POSITION
344
345 DBFF 20 70 DC JSR _CVFRO , CONVERT FRO TO ASCII CHAR IN LBUFF
346
347 D902 20 A4 DC JSR _FNZERO , FIND LAST NON-ZERO CHARACTER
348 D905 09 00 ORA #$80 , TURN ON HIGH ORDER BIT FOR POINT END
349 D907 9D 80 05 STA LBUFF,X , STORE IT BACK IN BUFFER
350
351 D90A AD 80 05 LDA LBUFF , GET 1ST CHAR IN LBUFF
352 D90D C9 2E CMP #' ' , IS IT DECIMAL?
353 D90F F0 03 BEQ _FN6 , BR IF YES
354 D911 4C 88 D9 JMP _FN5 , ELSE JUMP
355 D914 _FN6
356 D914 20 C1 DC JSR _DECINB , DECIMAL INBUFF
357 D917 4C 9C D9 JMP _FN4 , DO FINAL ADJUSTMENT

```

ERR LINE ADDR B1 B2 B3 B4

FLOATING POINT

PAGE 11

358

PAGE

359

EXPONENT IS ZERO - # IS ZERO

360

361

362

363 D91A

EXPO

364 D91A A9 B0

LDA #B0+B30 ; GET ASCII 0 WITH MSB = 1

365 D91C B0 B0 05

STA LBUFF ; PUT IN BUFFER

366 D91F 60

RTS


```

421
422 D95E A9 00 LDA #0 ; SUBTRACT FROM 0 TO COMPLEMENT
423 D960 35 SEC ; X
424 D961 E5 ED SBC EEXP ; X
425 D963 B5 ED STA EEXP
426
427 D965 A9 2D LDA #1 ; GET A MINUS
428 D967 D0 02 BNE _EF2
429
430 D969 _EPL
431 D969 A9 2B LDA #'+' ; GET A PLUS
432 D96B _EF2
433 D96B 20 9F DC JSR _STCHAR ; STORE A CHARACTER
434
435 D96E A2 00 LDX #0 ; SET COUNTER FOR # OF TENS
436 D970 A5 ED LDA EEXP ; GET EXPONENT
437
438 D972 _EF3
439 D972 38 SEC
440 D973 E9 0A SBC #10 ; SUBTRACT 10
441 D975 90 03 BCC _EF4 ; IF < 0, BRANCH
442 D977 E8 INX ; INCR # OF 10'S
443 D978 D0 F8 BNE _EF3 ; BR UNCONDITIONAL
444
445 D97A _EF4
446 D97A 18 CLC ; ADD BACK IN 10
447 D97B 69 0A ADC #10 ; X
448 D97D 48 PHA ; SAVE
449
450 D97E 8A TXA ; GET # OF 10'S
451 D97F 20 9D DC JSR _STNUM ; PUT 10'S IN EXP IN BUFFER
452 D982 68 PLA ; GET REMAINDER
453 D983 09 80 ORA #$80 ; TURN ON HIGH ORDER BIT FOR PRINT END
454 D985 20 9D DC JSR _STNUM ; PUT IN BUFFER
455
456
457
458 D988 _FN5
459 D988 AD 80 05 LDA LBUFF ; GET 1ST BYTE IN LBUFF (OUTPUT)
460 D98B C9 3D CMP #'0' ; IS IT ASCII 0?
461 D98D D0 0D BNE _FN4 ; IF NOT BR
462
463
464
465 D98F 18 CLC ; ADD 1 TO INBUFF
466 D990 A5 F3 LDA INBUFF ; X
467 D992 69 01 ADC #1 ; X
468 D994 B5 F3 STA INBUFF ; X
469 D996 A5 F4 LDA INBUFF+1 ; X
470 D998 69 00 ADC #0 ; X
471 D99A B5 F4 STA INBUFF+1 ; X
472 D99C _FN4
473 D99C A5 D4 LDA FRD ; GET EXPONENT OF #
474 D99E 10 09 BPL _FADONE ; IF SIGN +, WE ARE DONE

```

ERR LINE ADDR B1 B2 B3 B4

FLOATING POINT

PAGE 14

475

476 D9A0 20 C1 DC

477 D9A3 A0 00

478 D9A5 A7 2D

479 D9A7 91 F3

480

481 D9A9

482 D9A9 60

JSR

_DECINB

; DECR INBUFF

LDY

#0

; GET INDEX

LDA

#'-

; GET ASCII -

STA

(INBUFF),Y

; SAVE - IN BUFFER

_FADONE

RTS

```

483
484
485
486
487
488
489
490
491
492 D7AA CVIFP
493 D7AA IFP
494
495
496
497 D9AA A5 D4
498 D9AC 85 FB
499 D9AE A5 D5
500 D9B0 85 F7
501
502 D9B2 20 44 DA
503 D9B5 FB
504
505
506
507 D9B6 A0 10
508 D9B8
509 D9B8 06 FB
510 D9BA 26 F7
511
512
513 D9BC A2 03
514 D9BE
515
516
517
518 D9BE B5 D4
519 D9C0 75 D4
520 D9C2 95 D4
521 D9C4 CA
522 D9C5 D0 F7
523
524 D9C7 88
525 D9C8 D0 EE
526 D9CA D8
527
528
529
530 D9CB A9 42
531 D9CD 85 D4
532
533 D9CF 4C 00 DC
534

```

PAGE

IFP - CONVERT INTEGER TO FLOATING POINT

ON ENTRY FRO - CONTAINS INTEGER

ON EXIT FRO - CONTAINS FLOATING POINT #

MOVE INTEGER AND REVERSE BYTES

DO THE CONVERT

IFP1

CARRY NOW SET IF THERE WAS A BIT

IFP2

DOUBLE # AND ADD IN 1 IF CARRY SET

SET EXPONENT

NORMALIZE

LDY #16 ; GET # BITS IN INTEGER

ASL ZTEMP4+1 ; SHIFT LEFT INTEGER LOW

ROL ZTEMP4 ; SHIFT LEFT INTEGER HIGH

LDA FRO,X ; GET BYTE

ADC FRO,X ; DOUBLE (ADDING IN CARRY FROM SHIFT)

STA FRO,X ; SAVE

DEX ; DECREMENT COUNT OF FRO BYTES

BNE _IFP2 ; IF MORE TO DO, DO IT

DEY ; DECR COUNT OF INTEGER DIGITS

BNE _IFP1 ; IF MORE TO DO, DO IT

CLD ; CLEAR DECIMAL MODE

LDA #42 ; INDICATE DECIMAL AFTER LAST DIGIT

STA FRO ; STORE EXPONENT

JMP NORM ; NORMALIZE

[illegible]

FLOATING POINT

PAGE 14

```

589 D9FA 00 48          BCS  _ERVAL      ; N TO BIG
590 D9FC 00 5A DA      JSR  _LEFT      ; MULT BY +2 (NOW + 8 IN ZTEMP4)
591 D9FF 00 41          BCS  _ERVAL      ; BR IF N TO BIG
592
593 DA01 18            CLC              ; ADD IN * 2 TO = *10
594 DA02 A5 F6          LDA  ZTEMP4+1    ; X
595 DA04 65 F4          ADC  ZTEMP3+1    ; X
596 DA06 85 F8          STA  ZTEMP4+1    ; X
597 DA08 A5 F7          LDA  ZTEMP4      ; X
598 DA0A 65 F9          ADC  ZTEMP3      ; X
599 DA0C 85 F7          STA  ZTEMP4      ; X
600 DA0E 80 32          BCS  _ERVAL      ; IF CARRY SET ERROR
601
602
603
604
605 DA10 20 B9 DC      JSR  _GETDIG     ; GET DIGIT IN A
606 DA13 18            CLC              ;
607 DA14 65 F8          ADC  ZTEMP4+1    ; ADD IN DIGIT
608 DA16 85 F8          STA  ZTEMP4+1    ; X
609 DA18 A5 F7          LDA  ZTEMP4      ; X
610 DA1A 69 00          ADC  #0          ; X
611 DA1C 80 24          BCS  _ERVAL      ; BR IF OVERFLOW
612 DA1E 65 F7          STA  ZTEMP4      ; X
613
614 DA20 C6 F5          DEC  ZTEMP1      ; DEC COUNTER OF DIGITS TO DO
615 DA22 D0 C6          BNE  _FPI1      ; IF MORE TO DO, DO IT
616
617
618
619 DA24
620 DA24 20 B9 DC      JSR  _GETDIG     ; GET NEXT DIGIT IN A
621 DA27 C9 05          CMP  #5          ; IS DIGIT <5?
622 DA29 90 0D          BCC  _NR        ; IF YES, DON'T ROUND
623 DA2B 18            CLC              ; ADD IN 1 TO ROUND
624 DA2C A5 F8          LDA  ZTEMP4+1    ; X
625 DA2E 69 01          ADC  #1          ; X
626 DA30 85 F8          STA  ZTEMP4+1    ; X
627 DA32 A5 F7          LDA  ZTEMP4      ; X
628 DA34 69 00          ADC  #0          ; X
629 DA36 85 F7          STA  ZTEMP4      ; X
630
631
632
633 DA38
634 DA38 A5 F8          LDA  ZTEMP4+1    ; GET INTEGER LOW
635 DA3A 85 D4          STA  FRO         ; SAVE
636 DA3C A5 F7          LDA  ZTEMP4      ; GET INTEGER HIGH
637 DA3E 85 D5          STA  FRO+1      ; SAVE
638
639 DA40 18            CLC              ; CLEAR CC FOR GOOD RETURN
640 DA41 60            RTS
641
642

```

641
642

ERR LINE ADDR B1 B2 B3 B4

FLOATING POINT

PAGE 18

643 DA42

ERVAL

644 DA42 38

SEC

SET CARRY FOR ERROR RETURN

645 DA43 60

RTS

```

546                                     PAGE
547
548                                     ZFRO - ZERO FRO
549
550                                     ZF1 - ZERO 6 BYTES AT LOC X
551
552                                     ZXY - ZERO PAGE ZERO LOC X FOR LENGTH Y
553
554
555 DA44 ZFRO
556 DA44 A2 D4 LDX #FRO ; GET POINTER TO FRI
557
558 DA46 ZF1
559 DA46 A0 06 LDY #6 ; GET # OF BYTES TO CLEAR
560 DA48 ZXY
561 DA48 A9 00 LDA #0 ; CLEAR A
562 DA4A _ZF2
563 DA4A 95 00 STA 0/X ; CLEAR A BYTE
564 DA4C E6 INX ; POINT TO NEXT BYTE
565 DA4D 8B DEY ; DEC COUNTER
566 DA4E 00 FA BNE _ZF2 ; LOOP
567 DA50 60 RTS
568
569
570
571
572 INTLBF - INIT LBUFF INTO INBUFF
573
574 DA51 INTLBF
575 DA51 A9 05 LDA #.HIGH.LBUFF
576 DA53 85 F4 STA INBUFF+1
577 DA55 A9 80 LDA #.LOW.LBUFF
578 DA57 85 F3 STA INBUFF
579 DA59 60 RTS

```


ERR LINE ADDR B1 B2 B3 B4

FLOATING POINT

PAGE 20

680

PAGE

681

682

1

_ILSHFT - SHIFT INTEGER IN ZTEMP4 LEFT ONCE

683

1

684 DASA

ILSHFT

685 DASA

_ILSHFT

686 DASA 18

CLC

1 CLEAR CARRY

687 DASA 26 F8

ROL

ZTEMP4+1

1 SHIFT LOW

688 DASA 26 F7

ROL

ZTEMP4

1 SHIFT HIGH

689 DASA 60

RTS

```

690                                     PAGE    FLOATING POINT ROUTINES
691
692                                     FADD - FLOATING POINT ADD ROUTINE
693                                     ADDS VALUES IN FRO AND FR1
694
695                                     ON ENTRY   FRO & FR1 - CONTAIN # TO ADD
696
697                                     ON EXIT    FRO - RESULT
698
699
700                                     FSUB - FLOATING POINT SUBTRACT ROUTINE
701                                     SUBTRACTS FR1 FROM FRO
702
703                                     ON ENTRY   FRO & FR1 - CONTAIN # TO SUBTRACT
704
705                                     ON EXIT    FRO - RESULT
706
707
708                                     BOTH RETURN WITH CC SET
709                                     CARRY SET IF ERROR
710                                     CARRY CLEAR IF NO ERROR
711
712 DA60      FSUB
713 DA60      A5 E0      LDA      FR1          ; GET EXPONENT OF FR1
714 DA62      49 B0      EOR      #$80        ; CHANGE SIGN OF MANTISSA
715 DA64      85 E0      STA      FR1          ; SAVE EXPONENT
716
717
718
719 DA66      FADD
720 DA66      _FRADD
721 DA66      A5 E0      LDA      FR1          ; GET EXPONENT FR1
722 DA68      29 7F      AND      #$7F        ; TURN OFF MANTISSA SIGN BIT
723 DA6A      85 F7      STA      ZTEMP4      ; SAVE TEMPORARILY
724 DA6C      A5 D4      LDA      FRO         ; GET EXPONENT FRO
725 DA6E      29 7F      AND      #$7F        ; TURN OFF MANTISSA SIGN BIT
726 DA70      38        SEC                  ; CLEAR CARRY
727 DA71      E5 F7      SBC      ZTEMP4      ; SUB EXPONENTS
728 DA73      10 10      BPL      _NSWAP      ; IF EXP(FRO) >= EXP(FR1), NO SWAP
729
730                                     SWAP FRO AND FR1
731
732 DA75      A2 05      LDX      #FMPREC     ; GET INDEX
733
734 DA77      _SWAP
735 DA77      B5 D4      LDA      FRO,X       ; GET BYTE FROM FRO
736 DA79      B4 E0      LDY      FR1,X       ; GET BYTE FROM FR1
737 DA7B      95 E0      STA      FR1,X       ; PUT FRO BYTE IN FR1
738 DA7D      98        TYA                  ; GET FR1 BYTE
739 DA7E      95 D4      STA      FRO,X       ; PUT FR1 BYTE IN FRO
740 DA80      CA        DEX                  ; DEC INDEX
741 DA81      10 F4      BPL      _SWAP      ; IF MORE TO DO, GO SWAP
742 DA83      30 E1      BMI      _FRADD     ; UNCONDITIONAL
743

```

```

744 DAB3
745 DAB5 F0 07
746 DAB7 C9 05
747 DAB9 BD 19
748
749
750 DABB 20 3E DC
751
752
753
754 DABE
755 DABE F8
756 DABF A3 D4
757 DAF1 43 E0
758 DAF3 30 1E
759
760
761
762
763 DAF5 A2 04
764 DAF7 1B
765 DAF9
766 DAFB B5 D5
767 DAF4 75 E1
768 DAF6 95 D5
769 DAF8 CA
770 DAF0 10 F7
771
772 DAA1 D8
773 DAA2 00 03
774 DAA4
775 DAA4 4C 00 DC
776
777
778
779 DAA7
780 DAA7 A9 01
781 DAA9 20 3A DC
782
783 DAAC A9 01
784 DAAE B5 D5
785 DABD 4C 00 DC
786
787
788
789 DAB3
790 DAB3 A2 04
791 DAB5 3B
792
793 DAB6
794 DAB6 B5 D5
795 DAB8 F5 E1
796 DABA 95 D5
797 DABC CA
    
```

```

NSWAP
    BEQ     _NALIGN      ; IF DIFFERENCE = 0, ALREADY ALIGNED
    CMP     #FMPREC      ; IS DIFFERENCE < W OF BYTES IN MANTISSA?
    BCS     _ADDEND      ; IF NOT, HAVE RESULT IN FRO

    JSR     RSHFT1       ; SHIFT TO ALIGN

    TEST    FOR LIKE SIGN OF MANTISSA

    _NALIGN
    SED     FRO          ; SET DECIMAL MODE
    LDA     FRO          ; GET FRO EXPONENT
    EDR     FR1          ; EDR WITH FR1 EXPONENT
    RMI     _SUB         ; IF SIGNS DIFFERENT - SUBTRACT
ELSE ADD
    ADD     FRO & FR1

    LDY     #FMPREC-1    ; GET POINTER FOR LAST BYTE TO ADD
    CLC
_ADD1
    LDA     FROM,X       ; GET BYTE OF FRO
    ADC     FR1M,X       ; ADD IN BYTE OF FR1
    STA     FROM,X       ; STORE
    DEX
    BPL     _ADD1        ; ADD NEXT BYTE

    CLD
    BCS     _ADD2        ; CLEAR DECIMAL MODE
    IF THERE IS A CARRY, DO IT
_ADDEND
    JMP     NORM         ; GO NORMALIZE

    ADD     IN FIND CARRY

_ADD2
    LDA     #1           ; GET 1 TIMES TO SHIFT
    JSR     RSHFT0       ; GO SHIFT

    LDA     #01          ; GET CARRY
    STA     FROM         ; ADD IN CARRY
    JMP     NORM

    SUBTRACT FR1 FROM FRO
_SUB
    LDY     #FMPREC-1    ; GET POINTER TO LAST BYTE TO SUB
    BEC
_SUB1
    LDA     FROM,X       ; GET FRO BYTE
    SBC     FR1M,X       ; SUB FR1 BYTE
    STA     FROM,X       ; STORE
    DEX
    
```

```

798 DABD 10 F7          BPL      _SUB1      ; SUB NEXT BYTE
799
800 DABF 90 04          BCC      _SUB2      ; IF THERE IS A BORROW DO IT
801 DAC1 D8            CLD              ; CLEAR DECIMAL MODE
802 DAC2 4C 00 DC      JMP       NORM
803
804                      ;
805                      ; TAKE COMPLIMENT SIGN
806
806 DAC5              _SUB2
807 DAC5 A5 D4          LDA      FRO        ; GET EXPONENT
808 DAC7 49 80          EOR       #$80      ; CHANGE SIGN OF MANTISSA
809 DAC9 B5 D4          STA      FRO        ; PUT IT BACK
810
811                      ;
812                      ; COMPLIMENT MANTISSA
813
813 DACB 38            SEC              ; SET CARRY
814 DACC A2 04          LDX       #FMPREC-1 ; GET INDEX COUNTER
815 DACE              _SUB3
816 DACE A9 00          LDA      #0        ; GET ZERO
817 DAD0 F5 D5          SBC      FROM, X    ; COMPLEMENT BYTE
818 DAD2 95 D5          STA      FROM, X    ; STORE
819 DAD4 CA            DEX              ; MORE TO DO
820 DAD5 10 F7          BPL      _SUB3      ; BR IF YES
821
822 DAD7 D8            CLD              ; CLEAR DECIMAL MODE
823 DAD8 4C 00 DC      JMP       NORM      ; GO NORMALIZE

```



```

824                                     PAGE
825
826                                     FMULT - MULTIPLY FRO BY FR1
827
828                                     ON ENTRY  # ARE IN FRO AND FR1
829
830                                     ON EXIT   FRO - CONTAINS PRODUCT
831                                     RETURN WITH CC SET
832                                     CARRY SET IF ERROR
833                                     CARRY CLEAR IF NO ERROR
834
835
836
837 DADB                                FMUL
838
839                                     SET UP EXPONENT
840
841 DADB A5 D4                          LDA    FRO          ; GET EXP FRO
842 DADD F0 45                          BEQ    MEND3        ; IF = 0, DONE
843 DADF A5 E0                          LDA    FR1          ; GET FR1 EXP
844 DAE1 F0 3E                          BEQ    MEND2        ; IF = 0, ANSWER = 0
845
846 DAE3 20 CF DC                      JSR    MDESUP        ; DO COMMON SET FOR EXPONENT
847 DAE6 3B                            SEC                ; SET CARRY
848 DAE7 E9 40                          SBC    ##40         ; SUB EXCESS 64
849 DAE9 38                            SEC                ; SET CARRY TO ADD 1
850 DAEA 65 E0                          ADC    FR1          ; ADD 1 + FR1 EXP TO FRO EXP
851 DAEC 30 38                          BMI    _EROV        ; IF - THEN OVERFLOW
852
853                                     FINISH MULTIPLY SET UP
854
855 DAEE 20 E0 DC                      JSR    MDSUP        ; DO SET UP COMMON TO DIVIDE
856
857
858                                     DO THE MULTIPLY
859
860 DAF1                                _FRM
861
862                                     GET # OF TIMES TO ADD IN MULTIPLICAND
863
864 DAF1 A5 DF                          LDA    FRE+FMPPREC ; GET LAST BYTE OF FRE
865 DAF3 29 OF                          AND    ##OF        ; AND OUT HIGH ORDER NIBBLE
866 DAF5 85 F6                          STA    ZTEMP1+1    ; SET COUNTER FOR LOOP CONTROL
867
868                                     ADD IN FR1
869
870 DAF7                                _FRM1
871 DAF7 06 F6                          DEC    ZTEMP1+1    ; DEC MULT COUNTER
872 DAF9 30 06                          BMI    _FRM2        ; IF - THIS LOOP DONE
873 DAFB 20 01 DD                      JSR    FRA10        ; ADD FR1 TO FRO (6 BYTES)
874 DAFE 4C F7 DA                      JMP     _FRM1      ; REPEAT
875
876                                     GET # OF TIMES TO ADD IN MULTIPLICAND * 10
877

```

```

878 DB01          _FRM2      LDA    FRE+EMPREC    GET LAST BYTE OF FRE
879 DB01 A3 DF      LSR     LSR     SHIFT OUT LOW ORDER NIBBLE
880 DB03 4A        LSR     LSR     X
881 DB04 4A        LSR     LSR     X
882 DB05 4A        LSR     LSR     X
883 DB05 4A        LSR     LSR     X
884 DB07 85 F6     STA     ZTEMP1+1    SAVE AS COUNTER
885
886          ADD IN FR2
887
888 DB09          _FRM3
889 DB09 C4 F5     DEC     ZTEMP1+1    DECREMENT COUNTER
890 DB08 3D 06     BNE     _NXTB      IF -, DO NEXT BYTE
891 DB0D 2D 05 DD   JSR     FRA20     ADD FR2 TO FRO (5 BYTES)
892 DB10 4C 09 DB   JMP     _FRM3     REPEAT
893
894          SET UP FOR NEXT SET OF ADDS
895
896 DB13          _NXTB
897
898          SHIFT FRO/FRE RIGHT ONE BYTE
899          (THEY ARE CONTIGUOUS)
900
901 DB13 2D 62 DC   JSR     RSHFOE     SHIFT FRO/FRE RIGHT
902
903          TEST FOR # OF BYTES SHIFTED
904
905 DB16 C6 F5     DEC     ZTEMP1    DECREMENT LOOP CONTROL
906 DB16 0D D7     BNE     _FRM      IF MORE ADDS TO DO, DO IT
907
908          SET EXPONENT
909
910 DB1A          MDEND
911 DB1A A5 ED     LDA     EEXP        GET EXPONENT
912 DB1C 85 D4     STA     FRO        STORE AS FRO EXP
913
914          MEND1
915 DB1E 4C 04 DC   JMP     NORM1     NORMALIZE
916
917          MEND2
918
919          MEND3
920 DB21 2D 44 DA   JSR     ZFRO      CLEAR FRO
921 DB24 1B        CLC              CLEAR CARRY FOR GOOD RTN
922 DB24 80        RTS
923
924          EROV
925
926 DB26 38        SEC              SET CARRY FOR ERROR ROUTINE
927 DB27 60        RTS              RETURN
928
929
930

```

```

931      PAGE
932
933      FPDIV - FLOATING POINT DIVIDE
934
935      ON ENTRY   FRO - DIVIDEND
936                FR1 - DIVISOR
937
938      ON EXIT    FRO - QUOTIENT
939
940      RETURNS WITH CC SET
941                CARRY CLEAR - ERROR
942                CARRY SET - NO ERROR
943
944
945      DB28      FDIV
946
947      DO DIVIDE SET YUP
948
949      DB28      A5 E0      LDA      FR1          ; GET FR1 EXP
950      DB2A      F0 FA      BEQ      __EROV        ; IF =0, THEN OVERFLOW
951      DB2C      A5 D4      LDA      FRO          ; GET EXPONENT FRO
952      DB2E      F0 F4      BEQ      MEND3        ; IF = 0, THEN DONE
953
954      DB30      20 CF DC    JSR      MDESUP        ; DO COMMON PART OF EXP SET UP
955
956      DB33      38         SEC
957      DB34      E5 E0      SBC      FR1          ; SUB FR1 EXP FROM FRO EX
958      DB36      18         CLC
959      DB37      69 40      ADC      ##40         ; ADD IN EXCESS 64
960      DB39      30 EB      BMI      __EROV        ; IF MINUS THEN OVERFLOW
961
962      DB3B      20 E0 DC    JSR      MDSUP        ; DO SETUP COMMON FOR MULT
963      DB3E      E6 F5      INC      ZTEMP1       ; LOOP 1 MORE TIME FOR DIVIDE
964      DB40      4C 4E DB    JMP      _FRD1        ; SKIP SHIFT 1ST TIME THROUGH
965
966      OOD9      GTEMP      EQU      FRO+FMPPREC
967      DB43      _NEXTQ
968
969      SHIFT FRO/FRE LEFT ONE BYTE
970                (THEY ARE CONTIGUOUS)
971
972      DB43      A2 00      LDX      #0          ; GET POINTER TO BYTE TO MOVE
973      DB45      _NEXTQ1
974      DB45      B5 D5      LDA      FRO+1,X      ; GET BYTE
975      DB47      95 D4      STA      FRO,X        ; MOVE IT LEFT ONE BYTE
976
977      DB49      EB         INX                  ; POINT TO NEXT BYTE
978      DB4A      E0 0C      CPX      #FMPPREC*2+2 ; HAVE WE DONE THEM ALL?
979      DB4C      D0 F7      BNE      _NEXTQ1      ; IF NOT, BRANCH
980
981      DO DIVIDE
982
983      DB4E      _FRD1
984

```

```

985
986
987
988 DB4E AQ 05
989 DB50 38
990 DB51 F8
991 DB52
992 DB5C B9 DA 00
993 DB5D F9 E6 00
994 DB5E 99 DA 00
995 DB5F 88
996 DB60 10 F4
997 DB61 D8
998
999 DB6F 90 04
1000
1001 DB61 E6 D9
1002
1003 DB63 D0 E9
1004
1005
1006
1007 DB65
1008 DB65 20 OF DD
1009
1010
1011
1012 DB68 06 D9
1013 DB6A 06 D9
1014 DB6C 06 D9
1015 DB6E 06 D9
1016 DB70
1017
1018
1019
1020
1021 DB70 AQ 05
1022 DB72 38
1023 DB73 F8
1024 DB74
1025 DB74 B9 DA 00
1026 DB77 F9 E0 00
1027 DB7A 99 DA 00
1028 DB7D 88
1029 DB7E 10 F4
1030 DB80 D8
1031
1032 DB81 90 04
1033
1034 DB83 E6 D9
1035
1036 DB85 D0 E9
1037
1038
1039

```

SUBTRACT FR2 (DIVISOR * 2) FROM FRE (DIVIDEND)

```

LDY #FMPREC ; SET LOOP CONTROL
SEC ; SET CARRY
SED ; SET DECIMAL MODE
_FRS2
LDA FRE,Y ; GET A BYTE FROM FRE
SBC FR2,Y ; SUB FR2
STA FRE,Y ; STORE RESULT
DEY ; DEC COUNTER
BPL _FRS2 ; BR IF MORE TO DO
CLD ; CLEAR DECIMAL MODE
BCC _FAIL ; IF RESULT < 0 (FRE < FR2) BR
INC QTEMP ; INCR # TIMES SUB (QUOTIENT)
BNE _FRD1 ; SUB AGAIN
SUBTRACT OF FR2 DIDN'T GO
_FAIL
JSR FRA2E ; ADD FR2 BACK TO FRE
SHIFT LAST BYTE OF QUOTIENT ONE NIBBLE LEFT
ASL QTEMP ; SHIFT 4 BITS LEFT
ASL QTEMP ; X
ASL QTEMP ; X
ASL QTEMP ; X
_FRD2
SUBTRACT FR1 (DIVISOR) FROM FRE (DIVIDEND)
LDY #FMPREC ; SET LOOP CONTROL
SEC ; SET CARRY
SED ; SET DECIMAL MODE
_FRS1
LDA FRE,Y ; GET A BYTE FROM FRE
SBC FR1,Y ; SUB FR1
STA FRE,Y ; STORE RESULT
DEY ; DEC COUNTER
BPL _FRS1 ; BR IF MORE TO DO
CLD ; CLEAR DECIMAL MODE
BCC _FAIL2 ; IF RESULT < 0 (FRE < FR1) BR
INC QTEMP ; INCR # TIMES SUB (QUOTIENT)
BNE _FRD2 ; SUB AGAIN
SUBTRACT OF FR1 DIDN'T GO

```


ERR LINE ADDR B1 B2 B3 B4

FLOATING POINT ROUTINES

PAGE 28

1039

1040 DB87

1041 DB87 20 09 D0

1042

1043 DB8A 06 F5

1044 DB8C 00 E5

1045

1046 DB8E 20 62 DC

1047 DB91 40 1A DB

FAIL2

JBR

FRA1E

/ ADD FR1 BACK TO FRO

DEC

ZTEMP1

/ DEC LOOP CONTROL

BNE

_NXTQ

/ GET NEXT QUOTIENT BYTE

JSR

RSHFOE

/ SHIFT RIGHT FRO/FRE TO CLEAR EXP

JMP

MDEND

/ JOIN MULT END UP CODE


```

1096
1097
1098
1099
1100
1101
1102
1103 DBD0
1104 DBD8 AS F2
1105 DBD0 48
1106 DBDE 20 94 DB
1107 DDC1 90 1F
1108
1109 DBC3 C9 2E
1110 DBC5 F0 14
1111 DBC7 C9 2B
1112 DBC9 F0 07
1113 DBCB C9 2D
1114 DBCD F0 03
1115
1116
1117 DBCF
1118 DBCF 68
1119 DBD0 98
1120 DBD1 60
1121
1122
1123 DBD2
1124 DBD2 20 94 DB
1125 DBD5 90 0B
1126 DCD7 C9 2E
1127 DBD9 D0 F4
1128 DBD8
1129 DBD8 20 94 DB
1130 DBDE 90 02
1131 DBE0 B0 ED
1132
1133
1134 DBE2
1135 DBE2 68
1136 DBE3 85 F2
1137 DBE5 18
1138 DBE6 60

```

```

PAGE
TSTCHAR - TEST TO SEE IF THIS CAN BE A NUMBER
ON EXIT CC - CARRY SET IF NOT A #
          CARRY CLEAR IF A #
TSTCHAR
LDA      CIX          ; GET INDEX
PHA      ; SAVE IT
JSR      _GETCHAR     ; GET CHAR
BCC      _RTPASS      ; IF = #8 RETURN PASS
CMP      #' '         ; IF = D P , OK SO FAR
BEQ      _TSTN
CMP      #'+'         ; IF = +8 OK SO FAR
BEQ      _TSTN1
CMP      #'-'         ; IF = -8 OK SO FAT
BEQ      _TSTN1
RTFAIL
PLA      ; CLEAR STACK
TSNFAIL SEC          ; SET FAIL
RTS
TSTN1
JSR      _GETCHAR     ; GET CHAR
BCC      _RTPASS      ; IF #, RETURN PASS
CMP      #' '         ; IS IT D.P
BNE      _RTFAIL      ; IF NOT, RETURN FAIL
TSTN
JSR      _GETCHAR     ; ELSE GET NEXT CHAR
BCC      _RTPASS      ; IF #, RETURN PASS
BCS      _RTFAIL      ; ELSE, RETURN FAIL
RTPASS
PLA      ; RESTORE CIX
STA      CIX          ; X
CLC      ; CLEAR CARRY
RTS      ; RETURN PASS

```

1139					PAGE	
1140						
1141					NIBSH0 - SHIFT FR0 ONE NIBBLE LEFT	
1142						
1143					NIBSH2 - SHIFT FR2 ONE NIBBLE LEFT	
1144						
1145	DBE7				NIBSH2	
1146	DBE7	A2 E7			LDX	#FR2+1 ; POINT TO 1ST MANTISSA BYTE OF FR2
1147	DBE9	D0 02			BNE	_NIB1
1148						
1149	DBE6				NIBSH0	
1150	DBE8	A2 05			LDX	#FROM ; POINT TO MANTISSA OF FR0
1151	DBED				_NIB1	
1152	DBED	A0 04			LDY	#4 ; GET # OF BITS TO SHIFT
1153	DBEF				_NIBS	
1154	DBEF	18			CLC	; CLEAR CARRY
1155	DBF0	36 04			ROL	4, X ; ROLL
1156	DBF2	36 03			ROL	3, X ; X
1157	DBF4	36 02			ROL	2, X ; X
1158	DBF6	36 01			ROL	1, X ; X
1159	DBF8	36 00			ROL	0, X ; X
1160	DBFA	26 EC			ROL	FRX ; SAVE SHIFTED NIBBLE
1161						
1162	DBFC	88			DEY	; DEC COUNT
1163	DBFD	D0 F0			BNE	_NIBS ; IF NOT = 0, REPEAT
1164	DBFF	60			RTS	


```

1163                                     PAGE
1164
1165                                     NORM - NORMMALIZE FLOATING POINT NUMBER
1166
1167
1168
1169 DC00 NORM
1170 DC00 A2 00 LDX #0 ; GET ZERO
1171 DC02 B5 DA STX FRO+FPREC ; FOR ADD NORM SHIFT IN A ZERO
1172 DC04
1173 DC04 A2 04 NORM1 LDX #FMPREC-1 ; GET MAX # OF BYTES TO SHIFT
1174 DC06 A5 D4 LDA FRO ; GET EXPONENT
1175 DC08 F0 2E BEQ _NDONE ; IF EXP=0, # =0
1176 DC0A
1177 DC0A A5 D5 _NORM LDA FROM ; GET 1ST BYTE OF MANTISSA
1178 DC0C D0 1A BNE _TSTBIG ; IF NOT = 0 THEN NO SHIFT
1179
1180 SHIFT 1 BYTE LEFT
1181
1182 DC0E A0 00 LDY #0 ; GET INDEX FOR 1ST MOVE BYTE
1183 DC10 _NSH
1184 DC10 B9 D6 00 LDA FROM+1,Y ; GET MOVE BYTE
1185 DC13 99 D5 00 STA FROM,Y ; STORE IT
1186 DC16 C8 INY
1187 DC17 C0 05 CPY #FMPREC ; ARE WE DONE
1188 DC19 90 F5 BCC _NSH ; IF NOT SHIFT AGAIN
1189
1190
1191 DECREMENT EXPONENT
1192
1193 DC1B C6 D4 DEC FRO ; DECRIMENT EXPONENT
1194
1195 DC1D CA DEX ; DEC COUNTER
1196 DC1E D0 EA BNE _NORM ; DO AGAIN IF NEEDED
1197
1198
1199
1200 DC20 A5 D5 LDA FROM ; IS MANTISSA STILL 0
1201 DC22 D0 04 BNE _TSTBIG ; IF NOT, SEE IF TO BIG
1202 DC24 85 D4 STA FRO ; ELSE ZERO EXP
1203 DC26 18 CLC
1204 DC27 60 RTS
1205
1206 DC28 _TSTBIG
1207 DC28 A5 D4 LDA FRO ; GET EXPONENT
1208 DC2A 29 7F AND #$7F ; AND OUT SIGN BIT
1209 DC2C C9 71 CMP #49+64 ; IS IT < 49+64?
1210 DC2E 90 01 BCC _TSTUND ; IF YES, TEST UNDERFLOW
1211 DC30 60 RTS ; SO RETURN
1212 DC31
1213 DC31 C9 0F _TSTUND CMP #-49+64 ; IS IT >=-49+64?
1214 DC33 B0 03 BCS _NDONE ; IF YES, WE ARE DIONE
1215 DC35 20 44 DA JSR ZFRO ; ELSE # IS ZERO
1216
1217 DC38 _NDONE
1218 DC38 18 CLC ; CLEAR CARRY FOR GOOD RETURN

```

ERR LINE ADDR B1 B2 B3 B4

FLOATING POINT ROUTINES

PAGE 33

1219 DC39 60

RTS

```

1220
1221
1222
1223
1224
1225
1226
1227
1228
1229 DC3A
1230 DC3A A2 D4
1231 DC3C D0 02
1232
1233 DC3E
1234 DC3E A2 E0
1235
1236 DC40
1237 DC40 86 F9
1238 DC42 85 F7
1239 DC44 85 F8
1240
1241 DC46
1242 DC46 A0 04
1243 DC48
1244 DC48 B5 04
1245 DC4A 95 05
1246 DC4C CA
1247 DC4D 88
1248 DC4E D0 F8
1249 DC50 A9 00
1250 DC52 95 05
1251
1252 DC54 A6 F9
1253 DC56 C6 F7
1254 DC58 D0 EC
1255
1256
1257
1258 DC5A 85 00
1259 DC5C 18
1260 DC5D 65 F8
1261 DC5F 95 00
1262 DC61 60

```

PAGE

RSHFT0 - SHIFT FRO RIGHT/INCR EXPONENT

RSHFT1 - SHIFT FR1 RIGHT/INCR EXPONENT

ON ENTRY A - # OF PLACES TO SHIFT

RSHFT0

LDX #FRO ; POINT TO FRO

BNE _RSH

RSHFT1

LDX #FR1 ; POINT TO FR1

_RSH

STX ZTEMP3 ; SAVE FR POINTER

STA ZTEMP4 ; SAVE # OF BYTES TO SHIFT AS COUNTER

STA ZTEMP4+1 ; SAVE FOR LATER

_RSH2

LDY #FMPREC-1 ; GET # OF BYTES TO MOVE EACH LOOP

_RSH1

LDA 4,X ; GET CHAR

STA 5,X ; STORE CHAR

DEX ; POINT TO NEXT BYTE

DEY ; DEC LOOP CONTROL

BNE _RSH1 ; IF MORE TO MOVE, DO IT

LDA #0 ; GET 1ST BYTE

STA 5,X ; STORE IT

LDX ZTEMP3 ; GET FR POINTER

DEC ZTEMP4 ; DO WE NEED TO SHIFT AGAIN?

BNE _RSH2 ; IF YES, DO IT

FIX EXPONENT

LDA 0,X ; GET EXPONENT

CLC

ADC ZTEMP4+1 ; SUB # OF SHIFTS

STA 0,X ; SAVE NEW EXPONENT

RTS

ERR LINE ADDR B1 B2 B3 B4

FLOATING POINT ROUTINES

PAGE 35

```
1263
1264
1265
1266
1267
1268 DC62
1269 DC62 A2 0A
1270
1271 DC64
1272 DC64 B5 D4
1273 DC66 95 D5
1274
1275 DC68 CA
1276 DC69 10 F9
1277 DC6B A9 00
1278 DC6D B5 D4
1279 DC6F 60
```

PAGE

RSHFOE- SHIFT FRO/FRE 1 BYTE RIGHT
(THEY ARE CONTIGUOUS)

RSHFOE

LDX #FMPREC*2 ; GET LOOP CONTROL

_NXTB1

LDA FRO,X ; GET A BYTE
STA FRO+1,X ; MOVE IT OVER 1

DEX ; DEC COUNTER
BPL _NXTB1 ; MOVE NEXT BYTE
LDA #0 ; GET ZERO
STA FRO ; SHIFT IT IN
RTS


```

1280
1281
1282
1283
1284
1285
1286
1287
1288 DC7D
1289 DC7D 85 F7
1290
1291 DC72 A2 00
1292 DC74 A0 00
1293
1294
1295
1296 DC76
1297 DC76 20 93 DC
1298 DC79
1299 DC79 38
1300 DC7A E9 D1
1301 DC7C 83 F7
1302
1303
1304
1305 DC7E 65 D5
1306 DC80 4A
1307 DC81 4A
1308 DC82 4A
1309 DC83 4A
1310 DC84 20 9D DC
1311
1312
1313
1314 DC87 05 D5
1315 DC89 29 0F
1316 DC8B 20 9D DC
1317
1318 DC8E EB
1319 DC8F ED 05
1320 DC91 90 E3
1321
1322
1323
1324 DC93
1325 DC93 A3 F7
1326 DC93 D0 05
1327 DC97 A9 2E
1328 DC99 20 9F DC
1329 DC9C
1330 DC9C 60

```

PAGE

_CVFRO - CONVERT EACH BYTE IN FRO TO 2 CHARACTERS
IN LBUFF

ON ENTRY A - DECIMAL POINT POSITION

_CVFRO

STA ZTEMP4 ; SAVE DECIMAL POSITION

LDX #0 ; SET INDEX INTO FROM

LDY #0 ; SET INDEX INTO OUTPUT LINE (LBUFF)

CONVERT A BYTE

_CVBYTE

JSR _TSTDP ; PUT IN D.P. NOW?

_CVB1

SEC ; DECREMENT DECIMAL POSITION

SBC #1 ; X

STA ZTEMP4 ; SAVE IT

DO 1ST DIGIT

LDA FROM, X ; GET FROM FRO

LSRA ; SHIFT OUT LOW ORDER BITS

LSRA ; TO GET 1ST DIGIT

LSRA ; X

LSRA ; X

JSR _STNUM ; GO PUT # IN BUFFER

DO SECOND DIGIT

LDA FROM, X ; GET NUMBER FROM FRO

AND #\$0F ; AND OUT HIGH ORDER BITS

JSR _STNUM ; GO PUT # IN BUFFER

INX ; INCR FRO POINTER

CPX #FMPREC ; DONE LAST FRO BYTE?

BCC _CVBYTE ; IF NOT, MORE TO DO

PUT IN DECIMAL POINT NOW?

_TSTDP

LDA ZTEMP4 ; GET DECIMAL POSITION

BNE _TST1 ; IF NOT = 0 RTN

LDA # ; GET ASCII DECIMAL POINT

JSR _TCHAR ; PUT D.P. IN BUFFER

_TST1

RTS

```
1331      _PAGE
1332
1333      _STNUM - PUT ASCII NUMBER IN LBUFF
1334
1335      ON ENTRY  A - DIGIT TO BE CONVERTED TO ASCII
1336                AND PUT IN LBUFF
1337                Y - INDEX IN LBUFF
1338
1339      _STCHAR - STORES CHAR IN A IN LBUFF
1340
1341      DC9D      _STNUM
1342      DC9D 09 30      ORA      ##30      ; CONVERT TO ASCII
1343      DC9F      _STCHAR
1344      DC9F 99 80 05      STA      LBUFF,Y      ; PUT IN LBUFF
1345      DCA2 08      INY      ; INCR LBUFF POINTER
1346      DCA3 50      RTS
```

```

1347                                     _PAGE
1348
1349                                     _FNZERO - FIND LAST NON-ZERO CHAR IN LBUFF
1350
1351                                     ON EXIT   A - LAST CHAR
1352                                               X - POINT TO LAST CHAR
1353
1354 DCA4                                     _FNZERO
1355 DCA4 A2 0A                               LDX   #10           ; POINT TO LAST CHAR IN LBUFF
1356
1357 DCA6                                     _FN3
1358 DCA6 00 00 05                           LDA   LBUFF, X       ; GET THE CHARACTER
1359 DCA9 09 2E                           CMP   #' '           ; IS IT DECIMAL?
1360 DCAB F0 07                           BEQ   _FN1           ; IF YES, BR
1361 DCA0 09 30                           CMP   #'0'           ; IS IT ZERO?
1362 DCAF D0 07                           BNE   _FN2           ; IF NOT, BR
1363 DCB1 0A                               DEX                     ; DECREMENT INDEX
1364 DCB2 D0 F2                           BNE   _FN3           ; UNCONDITIONAL BR
1365
1366 DCB4                                     _FN1
1367 DCB4 0A                               DEX                     ; DECREMENT BUFFER INDEX
1368 DCB5 00 00 05                           LDA   LBUFF, X       ; GET LAST CHAR
1369 DCB8
1370 DCB8 60                               _FN2
                                           RTS

```

```
1371                                     PAGE
1372
1373                                     _GETDIG - GET NEXT DIGIT FROM FRO
1374
1375                                     ON ENTRY   FRO - #
1376
1377                                     ON EXIT    A - DIGIT
1378
1379
1380 DCB9                                     _GETDIG
1381 DCB9 20 EB DB                               JSR     NIBSHO      ; SHIFT FRO LEFT ONE NIBBLE
1382
1383 DCBC A5 EC                               LDA     FRX        ; GET BYTE CONTAINING SHIFTED NIBBLE
1384 DCBE 29 OF                               AND     #$0F      ; AND OUT HIGH ORDER NIBBLE
1385 DCC0 60                               RTS
```


ERR LINE ADDR B1 B2 B3 B4

FLOATING POINT ROUTINES

PAGE 40

1386

1387

1388

1389

1390 DCC1

1391 DCC1 38

1392 DCC2 A5 F3

1393 DCC4 E9 01

1394 DCC6 85 F3

1395 DCC8 A5 F4

1396 DCCA E9 00

1397 DCCC 85 F4

1398 DCCF 6D

PAGE

_DECINB - DECREMENT INBUFF

_DECINB

```
SEC                                ; SUBTRACT ONE FROM INBUFF
LDA    INBUFF                    ; X
SEC     #1                        ; X
STA    INBUFF                    ; X
LDA     INBUFF+1                  ; X
SEC     #0                        ; X
STA     INBUFF+1                  ; X
RTS
```

```
1399          PAGE
1400
1401          MDESUP - COMMON SET UP FOR MULTFLY & DIVIDE EXPONENT
1402
1403          ON EXIT      FR1 - FR1 EXP WITH OUT SIGN
1404                      A - FRO EXP WITHOUT SIGN
1405                      FRSIGN - SIGN FOR QUOTIENT
1406
1407          DCCF          MDESUP
1408          DCCF  A5 D4    LDA      FRO          ; GET FRO EXPONENT
1409          DCD1  45 E0    EOR      FR1          ; GET FR1 EXPONENT
1410          DCD3  29 80    AND      #$80         ; AND OUT ALL BUT SIGN BIT
1411          DCD5  85 EE    STA      FRSIGN       ; SAVE SIGN
1412
1413          DCD7  06 E0    ASL      FR1          ; SHIFT OUT SIGN IN FR1 EXP
1414          DCD9  46 E0    LSR      FR1          ; RESTORE FR1 EXP WITHOUT SIGN
1415          DCDB  A5 D4    LDA      FRO          ; GET FRO EXP
1416          DCDD  29 7F    AND      #$7F         ; AND OUT SIGN BIT
1417          DCDF  60      RTS
```

1418

1419

1420

1421

1422

1423

1424

1425

1426 DCE0

1427 DCE0 05 EE

1428 DCE2 85 ED

1429 DCE4 A9 00

1430 DCE6 83 D4

1431 DCE8 83 E0

1432

1433

1434 DCEA 20 28 DD

1435

1436 DCE0 20 E7 DB

1437 DCF0 43 EC

1438 DCF2 2F 0F

1439 DCF4 B3 E6

1440

1441 DCF6 A9 05

1442 DCF8 85 F5

1443

1444 DCF A 20 34 DD

1445 DCF0 20 44 DA

1446

1447 DD00 60

1448

PAGE

MDSUP - COMMON SET UP FOR MULTIPLY AND DIVIDE

ON ENTRY A - EXPONENT

CC - SET BY ADD OR SUB TO GET A

MDSUP

DRA FRSIGN

STA EEXP

LDA #0

STA FRO

STA FR1

1 OR IN SIGN BIT

1 SAVE EXPONENT FOR LATER

1 CLEAR A

1 CLEAR FRO EXP

1 CLEAR FRO EXP

JSR MVER12

1 MOVE FR1 TO FR2

JSR NIBSH2

LDA FRX

AND #0F

STA FR2

1 SHIFT FR2 1 NIBBLE LEFT

1 GET SHIFTED NIBBLE

1 AND OUT HIGH ORDER NIBBLE

1 STORE TO FINISH SHIFT

LDA #FMPREC

STA ITEMP1

1 SET LOOP CONTROL

1 X

JSR MVER0E

JSR ZFRO

1 MOVE FRO TO FRE

1 CLEAR FRO

RTS

```

1449                                     PAGE
1450
1451                                     FRA10 - ADD FR1 TO FRO (6 BYTES)
1452
1453                                     FRA20 - ADD FR2 TO FRO (6 BYTES)
1454
1455                                     FRA1E - ADD FR1 TO FRE
1456
1457                                     FRA2E - ADD FR2 TO FRE
1458
1459 DD01 FRA10
1460 DD01 A2 D9 LDX #FRO+FMPREC ; POINT TO LAST BYTE OF SUM
1461 DD03 DO 06 BNE _F1
1462
1463 DD05 FRA20
1464 DD05 A2 D9 LDX #FRO+FMPREC
1465 DD07 DO 06 BNE _F2
1466
1467 DD09 FRA1E
1468 DD09 A2 DF LDX #FRE+FMPREC
1469 DD0B _F1
1470 DD0B A0 E5 LDY #FR1+FMPREC
1471 DD0D DO 04 BNE _FRA
1472 DD0F FRA2E
1473 DD0F A2 DF LDX #FRE+FMPREC
1474 DD11 _F2
1475 DD11 A0 EB LDY #FR2+FMPREC
1476
1477
1478 DD13 _FRA
1479 DD13 A9 05 LDA #FMPREC ; GET VALUE FOR LOOP CONTROL
1480 DD15 B5 F7 STA ZTEMP4 ; SET LOOP CONTROL
1481 DD17 18 CLC ; CLEAR CARRY
1482 DD18 F8 SED ; SET DECIMAL MODE
1483 DD19 _FRA1
1484 DD19 B5 00 LDA O,X ; GET 1ST BYTE OF
1485 DD1B 79 00 00 ADC O,Y ; ADD
1486 DD1E 95 00 STA O,X ; STORE
1487 DD20 CA DEX ; POINT TO NEXT BYTE
1488 DD21 B8 DEY ; POINT TO NEXT BYTE
1489 DD22 C6 F7 DEC ZTEMP4 ; DEC COUNTER
1490 DD24 10 F3 BPL _FRA1 ; IF MORE TO DO, DO IT
1491 DD26 D8 CLD ; CLEAR DECIMAL MODE
1492 DD27 60 RTS

```


ERR LINE ADDR B1 B2 B3 B4

FLOATING POINT ROUTINES

PAGE 49

1493

1494

1495

1496

1497 0028

1498 0028 A0 05

1499 002A

1500 002A 89 E0 00

1501 002D 99 E8 00

1502

1503 0030 89

1504 0031 10 F7

1505 0033 60

1506

1507

1508

1509

1510

1511

1512

1513 0034

1514 0034 A0 05

1515 0036

1516 0036 89 D4 00

1517 0039 99 DA 00

1518

1519 003C 89

1520 003D 10 F7

1521 003F 60

PAGE

MVFR12 - MOVE FR1 TO FR2

MVFR12

_MV2 LDY #FMPREC ; SET COUNTER

LDA FR1,Y ; GET A BYTE

STA FR2,Y ; STORE IT

DEY ; DEC COUNTER

BPL _MV2 ; IF MORE TO MOVE, DO IT

RTS

MVFR0E - MOVE FRO TO FRE

MVFR0E

_MV1 LDY #FMPREC

LDA FRO,Y

STA FRE,Y

DEY

BPL _MV1

RTS

```

1522                                     PAGE 'POLYNOMIAL EVALUATION'
1523                                     Y=A(0)+A(1)*X+A(2)*X**2+ . . . +A(N)*X**N,N>0
1524                                     *(( . . . (A(N)*X+A(N-1))*X+. . . +A(2))*X+A(1))*X+A(0)
1525                                     INPUT X IN FRO, N+1 IN A-REG
1526                                     REG (X,Y)->A(N)...A(0)
1527                                     OUTPUT Y IN FRO
1528                                     USES FPTR2, PLYCNT, PLYARG
1529                                     CALLS FSTOR, FMOVE, FLD1R, FADD, FMUL
1530 DD40 B6 FE PLYEVL STX FPTR2 ;SAVE POINTER TO COEFF'S
1531 DD42 B4 FF STY FPTR2+1
1532 DD44 B5 EF STA PLYCNT
1533 DD46 A2 E0 LDX # LOW.PLYARG
1534 DD48 A0 05 LDY #.HIGH.PLYARG
1535 DD4A 20 A7 DD JSR FSTOR ;SAVE ARG
1536 DD4D 20 B6 DD JSR FMOVE ;ARG->FR1
1537 DD50 A6 FE LDX FPTR2
1538 DD52 A4 FF LDY FPTR2+1
1539 DD54 20 89 DD JSR FLD0R ;COEF->FRO (INIT SUM)
1540 DD57 C6 EF DEC PLYCNT
1541 DD59 F0 2D BEQ PLYOUT ;DONE ?
1542 DD5B 20 DB DA PLYEV1 JSR FMUL ; SUM * ARG
1543 DD5E B0 28 BCS PLYOUT ; O'FLOW
1544 DD60 18 CLC
1545 DD61 A5 FE LDA FPTR2 ;BUMP COEF POINTER
1546 DD63 69 06 ADC #FPREC
1547 DD65 B5 FE STA FPTR2
1548 DD67 90 06 BCC PLYEV2
1549 DD69 A5 FF LDA FPTR2+1 ;ACROSS PAGE
1550 DD6B 69 00 ADC #0
1551 DD6D B5 FF STA FPTR2+1
1552 DD6F A6 FE PLYEV2 LDX FPTR2
1553 DD71 A4 FF LDY FPTR2+1
1554 DD73 20 98 DD JSR FLD1R ;GET NEXT COEF
1555 DD76 20 66 DA JSR FADD ;SUM*ARG + COEF
1556 DD79 B0 0D BCS PLYOUT ; O'FLOW
1557 DD7B C6 EF DEC PLYCNT
1558 DD7D F0 09 BEQ PLYOUT ;DONE ?
1559 DD7F A2 E0 LDX # LOW.PLYARG
1560 DD81 A0 05 LDY # HIGH.PLYARG
1561 DD83 20 98 DD JSR FLD1R ;GET ARG AGAIN
1562 DD86 30 D3 BMI PLYEV1 ; (=JMP)
1563 DD88 60 PLYOUT RTS

```

```

1564
1565
1566 DD89 85 FC
1567 DD8B 84 FD
1568 DD8D A0 05
1569 DD8F B1 FC
1570 DD91 99 D4 00
1571 DD94 86
1572 DD95 10 F8
1573 DD97 60
1574
1575
1576 DD98 86 FC
1577 DD9A 84 FD
1578 DD9C A0 05
1579 DD9E B1 FC
1580 DDA0 99 E0 00
1581 DDA3 88
1582 DDA4 10 F8
1583 DDA6 60
1584
1585
1586 DDA7 86 FC
1587 DDA9 84 FD
1588 DDAB A0 05
1589 DDAD B9 D4 00
1590 DDB0 91 FC
1591 DDB2 88
1592 DDB3 10 F8
1593 DDB5 60
1594
1595
1596
1597 DDB6
1598 DDB6 A2 05
1599 DDB8 B5 D4
1600 DDBA 95 E0
1601 DDBC CA
1602 DDBD 10 F9
1603 DDBF 60

```

PAGE 'FLOATING LOAD/STORE'
 LOAD FRO FROM (X,Y) X=LSB, Y=MSB, USES FLPTR (PGO)
 FLDOR STX FLPTR , SET FLPTR => (X,Y)
 STY FLPTR+1
 FLDOP LDY #FPREC-1 , # BYTES ENTER HERE W/FLPTR SET
 FLD01 LDA (FLPTR),Y , MOVE
 STA FRO,Y
 DEY
 BPL FLD01 , COUNT & LOOP
 RTS

LOAD FR1 FROM (X,Y) OR (FLPTR)
 FLD1R STX FLPTR , FLPTR=>(X,Y)
 STY FLPTR+1
 FLD1P LDY #FPREC-1 , # BYTES ENTER W/FLPTR SET
 FLD11 LDA (FLPTR),Y , MOVE
 STA FR1,Y
 DEY
 BPL FLD11 , COUNT & LOOP
 RTS

STORE FRO IN (X,Y) OR (FLPTR)
 FSTOR STX FLPTR
 STY FLPTR+1
 FSTOP LDY #FPREC-1 , ENTRY W/FLPTR SET
 FST01 LDA FRO,Y
 STA (FLPTR),Y
 DEY
 BPL FST01
 RTS

MOVE FRO TO FR1
 MVOT01
 FMOVE LDX #FPREC-1
 FMOVE1 LDA FRO,X
 STA FR1,X
 DEX
 BPL FMOVE1
 RTS

EXP3	STA	FR1+2,X	CLEAR REST OF MANTISSA
	DEY		
	BPL	EXP3	
	LDA	FR1	BACK TO EXPONENT
	CLC		
	ADC	#40	BIAS IT
	BCB	EXPERR	DOPS. IT'S TOO BIG
	BMI	EXPERR	
	STA	FR1	FR1 = 10**I
	JSR	FMUL	(10**I)*(10**F)
EXPSGN	LDA	SGNFLG	WAS ARG<0
	BPL	EXPOUT	NO-DONE
	JSR	FMOVE	YES-INVERT RESULT
	LDX	# LOW FONE	
	LDY	# HIGH FONE	
	JSR	FLDOR	
	JSR	FDIV	
EXPOUT	RTS		(PANT, PANT - FINISHED!!)
EXPERR	SEC		FLAG ERROR
	RTS		& QUIT
P10COF	BYTE	\$3D,\$17,\$94,\$19,0,0,0,0	0000179419
	BYTE	\$3D,\$57,\$33,\$05,0,0,0,0	0000573305
	BYTE	\$3E,\$05,\$54,\$76,\$62,0,0,0	0005547662
	BYTE	\$3E,\$32,\$19,\$62,\$27,0,0,0	0032176227
	BYTE	\$3F,\$01,\$68,\$60,\$30,\$36,0,0	0168603036
	BYTE	\$3F,\$07,\$32,\$03,\$27,\$41,0,0	0732032741
	BYTE	\$3F,\$25,\$43,\$34,\$56,\$75,0,0	2543345675
	BYTE	\$3F,\$66,\$27,\$37,\$30,\$50,0,0	6627373050
	BYTE	\$40,\$01,\$15,\$12,\$92,\$55,1,1	15129255
	BYTE	\$3F,\$99,\$99,\$99,\$99,\$99,0,0	9999999999
NPCOEFF	EGU	*(-P10COF)/FPREC	
LOG10E	BYTE	\$3F,\$43,\$42,\$94,\$48,\$19,0,0	LOG10(E)
FONE	BYTE	\$40,1,0,0,0,0,0,0	1 0

```

1703
1704 DE95 86 FE          XFORM STX      FPTR2
1705 DE97 84 FF          STY      FPTR2+1
1706 DE99 A2 E0          LDX      # LOW.PLYARG
1707 DE9B A0 05          LDY      # HIGH.PLYARG
1708 DE9D 20 A7 DD      JSR      FSTOR      ; STASH X IN PLYARG
1709 DEA0 A6 FE          LDX      FPTR2
1710 DEA2 A4 FF          LDY      FPTR2+1
1711 DEA4 20 98 DD      JSR      FLD1R
1712 DEA7 20 66 DA      JSR      FADD      ; X+C
1713 DEAA A2 E6          LDX      #. LOW.FPSCR
1714 DEAC A0 05          LDY      #. HIGH.FPSCR
1715 DEAE 20 A7 DD      JSR      FSTOR
1716 DEB1 A2 E0          LDX      #. LOW.PLYARG
1717 DEB3 A0 05          LDY      #. HIGH.PLYARG
1718 DEB5 20 89 DD      JSR      FLDOR
1719 DEB8 A6 FE          LDX      FPTR2
1720 DEBA A4 FF          LDY      FPTR2+1
1721 DEBC 20 98 DD      JSR      FLD1R
1722 DEBF 20 60 DA      JSR      FSUB      ; X-C
1723 DEC2 A2 E6          LDX      #. LOW.FPSCR
1724 DEC4 A0 05          LDY      #. HIGH.FPSCR
1725 DEC6 20 98 DD      JSR      FLD1R
1726 DEC9 20 28 DB      JSR      FDIV      ; (X-C)/(X+C) = Z
1727 DECC 60          RTS

```

```

1728
1729 DECD A9 01      LOG      LDA      #1      REMEMBER ENTRY POINT
1730 DECF D0 02      LOG1     BNE      LOGBTH
1731 DED1 A9 00      LOG10    LDA      #0      CLEAR FLAG
1732 DED3 85 F0      LOGBTH   STA      SGNFLG  USE SGNFLG FOR LOG/LOG10 MARKER
1733 DED5 A5 D4      LDA      FRO
1734 DED7 F0 05      BEQ      LOGERR  LOGCO>
1735 DED9 30 03      BMI      LOGERR
1736
1737      WE WANT X = F*(10**Y), 1<F<10
1738      10**Y HAS SAME EXP BYTE AS X
1739      & MANTISSA BYTE = 1 OR 10
1739 DEDB 4C F6 DF      LOG1     JMP      LOG1XX  MOVED CODE
1740 DEDE 38          LOGERR    SEC
1741 DEDF 60          RTS
1742 DEE0 E9 40      LOG1YY   SBC      #$40      RETURN FROM MOVED CODE
1743 DEE2 0A          ASLA
1744 DEE3 85 F1      STA      XFMFLG  REMEMBER Y
1745 DEE5 A5 D5      LDA      FRO+1
1746 DEE7 29 F0      AND      #$F0
1747 DEE9 D0 04      BNE      LOG2
1748 DEEB A9 01      LDA      #1
1749 DEED D0 04      BNE      LOG3
1750 DEEF E6 F1      LOG2     INC      XFMFLG  BUMP Y
1751 DEF1 A9 10      LDA      #$10
1752 DEF3 85 E1      LOG3     STA      FR1+1  SET UP MANTISSA
1753 DEF5 A2 04      LDX      #FPREC-2  CLEAR REST OF MANTISSA
1754 DEF7 A9 00      LDA      #0
1755 DEF9 95 E2      LOG4     STA      FR1+2,X
1756 DEFB CA          DEX
1757 DEFC 10 FB      BPL      LOG4
1758 DEFE 20 28 DB      JSR      FDIV      X = X/(10**Y) - S.B IN (1,10)
1759 DF01          FLOG10    EQU      *      LOG10(X), 1<X<=10
1760 DF01 A2 66      LDX      #.LOW.SQR10
1761 DF03 A0 DF      LDY      #.HIGH.SQR10
1762 DF05 20 95 DE      JSR      XFORM      Z = (X-C)/(X+C), C*C = 10
1763 DF08 A2 E6      LDX      #.LOW.FPSCR
1764 DF0A A0 05      LDY      #.HIGH.FPSCR
1765 DF0C 20 A7 DD      JSR      FSTOR      SAVE Z
1766 DF0F 20 B6 DD      JSR      FMOVE
1767 DF12 20 DB DA      JSR      FMUL      Z*Z
1768 DF15 A9 0A      LDA      #NLCDEF
1769 DF17 A2 72      LDX      #.LOW.LGCOEF
1770 DF19 A0 DF      LDY      #.HIGH.LGCOEF
1771 DF1B 20 40 DD      JSR      PLYEVL  P(Z*Z)
1772 DF1E A2 E6      LDX      #.LOW.FPSCR
1773 DF20 A0 05      LDY      #.HIGH.FPSCR
1774 DF22 20 98 DD      JSR      FLD1R
1775 DF25 20 DB DA      JSR      FMUL      Z*P(Z*Z)
1776 DF28 A2 6C      LDX      #.LOW.FHALF
1777 DF2A A0 DF      LDY      #.HIGH.FHALF
1778 DF2C 20 98 DD      JSR      FLD1R
1779 DF2F 20 66 DA      JSR      FADD      0.5 + Z*P(Z*Z)
1780 DF32 20 B6 DD      JSR      FMOVE
1781 DF35 A9 00      LDA      #0

```


1782	DF37	85 D8	STA	FRO+J	
1783	DF39	43 F1	LDA	XFMFLG	
1784	DF3B	85 D4	STA	FRO	
1785	DF3D	10 07	BPL	LOG6	
1786	DF3F	49 FF	EOR	#-1	FLIP SIGN
1787	DF41	18	CLC		
1788	DF42	69 01	ADC	#1	
1789	DF44	85 D4	STA	FRO	
1790	DF46		LOG6		
1791	DF48	20 AA D9	JSR	IFP	LEAVES FRI ALONE
1792	DF49	24 F1	BIT	XFMFLG	
1793	DF4B	10 06	BPL	LOG7	
1794	DF4D	A9 80	LDA	##80	FLIP AGAIN
1795	DF4F	05 D4	ORA	FRO	
1796	DF51	85 D4	STA	FRO	
1797	DF53		LOG7		
1798	DF53	20 66 DA	JSR	FADD	LOG(X) = LOG(X) + Y
1799	DF56		LOGOUT		
1800	DF56	A5 F0	LDA	SGNFLG	
1801	DF58	F0 0A	BEG	LOGDON	WAS LOG10, NOT LOG
1802	DF5A	A2 89	LDX	# LOW LOG10E; LOG(X)/LOG10(E)	
1803	DF5C	A0 DE	LDY	# HIGH LOG10E	
1804	DF5E	20 98 DD	JSR	FLD1R	
1805	DF61	20 28 DB	JSR	FDIV	
1806	DF64	18	LOGDON	CLC	
1807	DF65	60	RTS		
1808	DF66	40 03 16 22	SGR10	BYTE	\$40, \$03, \$16, \$22, \$77, \$66 ; SQUARE ROOT OF 10
1809	DF6A	77 66			
1810	DF6C	3F 50 00 00	FHALF	BYTE	\$3F, \$50, 0, 0, 0, 0 ; 0.5
1811	DF70	00 00			
1812	DF72	3F 49 15 57	LGCOEF	BYTE	\$3F, \$49, \$15, \$57, \$11, \$08 ; 0.4915571108
1813	DF76	11 08			
1814	DF78	BF 51 70 49		BYTE	\$BF, \$51, \$70, \$49, \$47, \$08 ; -0.5170494708
1815	DF7C	47 08			
1816	DF7E	3F 39 20 57		BYTE	\$3F, \$39, \$20, \$57, \$61, \$95 ; 0.3920576195
1817	DF82	61 95			
1818	DF84	BF 04 39 63		BYTE	\$BF, \$04, \$39, \$63, \$03, \$55 ; -0.0439630355
1819	DF88	03 55			
1820	DF8A	3F 10 09 30		BYTE	\$3F, \$10, \$09, \$30, \$12, \$64 ; 0.1009301264
1821	DF8E	12 64			
1822	DF90	3F 09 39 08		BYTE	\$3F, \$09, \$39, \$08, \$04, \$60 ; 0.0939080460
1823	DF94	04 60			
1824	DF96	3F 12 42 58		BYTE	\$3F, \$12, \$42, \$58, \$47, \$42 ; 0.1242584742
1825	DF9A	47 42			
1826	DF9C	3F 17 37 12		BYTE	\$3F, \$17, \$37, \$12, \$06, \$08 ; 0.1737120608
1827	DFA0	06 08			
1828	DFA2	3F 28 95 29		BYTE	\$3F, \$28, \$95, \$29, \$71, \$17 ; 0.28957117
1829	DFA6	71 17			
1830	DFA8	3F 86 85 88		BYTE	\$3F, \$86, \$85, \$88, \$96, \$44 ; 0.8685889644
1831	DFAC	96 44			
1832	000A		NLCOEF	EQU	*(-LGCOEF)/FPREC
1833	DFAE	3E 16 05 44	ATCOEF	BYTE	\$3E, \$16, \$05, \$44, \$49, 0 ; 0.0016054449
1834	DFB2	49 00			
1835	DFB4	BE 95 68 38		BYTE	\$BE, \$95, \$68, \$38, \$45, 0 ; -0.009568345

ERR LINE ADDR B1 B2 B3 B4

LOG10(X)

PAGE 52

1836	DFBB	45 00		
1837	DFBA	3F 02 68 79	. BYTE	\$3F, \$02, \$68, \$79, \$94, \$16 ; 0. 0268799416
1838	DFBE	74 16		
1839	DFC0	BF 04 92 78	. BYTE	\$BF, \$04, \$92, \$78, \$90, \$80 ; -0. 0492789080
1840	DFC4	90 80		
1841	DFC6	3F 07 03 15	. BYTE	\$3F, \$07, \$03, \$15, \$20, 0 ; 0. 0703152000
1842	DFCA	20 00		
1843	DFCC	BF 08 92 29	. BYTE	\$BF, \$08, \$92, \$29, \$12, \$44 ; -0. 0892291244
1844	DFD0	12 44		
1845	DFD2	3F 11 08 40	. BYTE	\$3F, \$11, \$08, \$40, \$09, \$11 ; 0. 1108400911
1846	DFD6	09 11		
1847	DFD8	BF 14 28 31	. BYTE	\$BF, \$14, \$28, \$31, \$56, \$04 ; -0. 1428315604
1848	DFDC	56 04		
1849	DFDE	3F 19 99 95	. BYTE	\$3F, \$19, \$99, \$98, \$77, \$44 ; 0. 1999987744
1850	DFF2	77 44		
1851	DFF4	BF 33 33 33	. BYTE	\$BF, \$33, \$33, \$33, \$31, \$13 ; -0. 3333333113
1852	DFF8	31 13		
1853	DFFA	3F 99 99 99	FP9S . BYTE	\$3F, \$99, \$99, \$99, \$99, \$99 ; 0. 9999999999
1854	DFF6	99 99		
1855	000B		NATCF EQU	*(-ATCOEF)/FPREC
1856	DFF0	3F 78 53 98	PIQV4 . BYTE	\$3F, \$78, \$53, \$98, \$16, \$34 ; PI/4 = ARCTAN(1.0)
1857	DFF4	16 34		
1858	DFF6	A5 D4	LOG1XX LDA	FRO ; MOVED CODE
1859	DFF8	85 E0	STA	FR1
1860	DFFA	38	SEC	
1861	DFFB	4C E0 DE	JMP	LOG1YY
1862	DFFE		. END	

ASSEMBLY ERRORS = 0

SYMBOL TABLE

AFP	DB00	ASCIN	DB00	ATCOEF	DFAE	CIX	DOF2
CVAFPP	DB00	CVFASC	DBE6	CVIFP	D9AA	DEGFLG	DOF8
DEWON	0006	DIGRT	00F1	EEXF	00ED	ESIGN	00EF
EXP	DDC0	EXP1	DE03	EXP10	DDCC	EXP2	DE20
EXP3	DE26	EXPERR	DE4B	EXPOUT	DE4A	EXPSGN	DE39
FADD	DA66	FASC	DBE6	FCHREF	00F0	FDIV	DB28
FHALF	DF6C	FLD01	DDBF	FLDOP	DDBD	FLDOR	DD89
FLD11	DD9E	FLD1P	DD9C	FLD1R	DD98	FLOG10	DF01
FLPTR	00FC	FMOVE	DDB6	FMOVE1	DDB8	FMPREC	0005
FMUL	DADB	FONE	DEBF	FP9S	DFEA	FPI	D9D2
FPORQ	DB00	FPREC	0006	FPSCR	05E6	FPSCR1	05EC
FPR2	00FE	FRO	00D4	FROM	00D5	FR1	00E0
FR1M	00E1	FR2	00E6	FRA10	DD01	FRA1E	DD09
FRA20	DD05	FRA2E	DD0F	FRE	00DA	FRSIGN	00EE
FRX	00EC	FSCR	05E6	FSCR1	05EC	FST01	DDAD
FSTOP	DDAB	FSTOR	DDA7	FSUB	DA60	IFP	D9AA
ILSHFT	DA5A	INBUFF	00F3	INTLBF	DA51	LBPR1	057E
LBPR2	057F	LBUFF	0580	LGCOEF	DF72	LOG	DECD
LOG1	DEDB	LQG10	DED1	LOG10E	DE89	LOG1XX	DF66
LOG1YY	DEE0	LOG2	DEEF	LOG3	DEF3	LOG4	DEF9
LOG6	DF46	LQG7	DF53	LOGBTH	DED3	LOGDON	DF64
LOGERR	DEDE	LOGOUT	DF56	MDEND	DB1A	MDESUP	DCCF
MDSUP	DCE0	MEMORY	0000	MEND1	DB1E	MEND2	DB21
MEND3	DB24	MVOTQ1	DDB6	MVFROE	DD34	MVFR12	DD28
NARG	0000	NATCF	000B	NIBSHO	DBEB	NIBSH2	DBE7
NLCOEf	000A	NORM	DC00	NORM1	DC04	NPCOEf	000A
NSIGN	00EE	PIOCOF	DE4D	PIOV4	DDF0	PLYARG	05E0
PLYCNT	00EF	PLYEV1	DD5B	PLYEV2	DD6F	PLYEVL	DD40
PLYOUT	DD8B	QTEMP	00D9	RADFLG	00FB	RADON	0000
RSHFOE	DC62	RSHFTO	DC3A	RSHFT1	DC3E	SGNFLG	00F0
SKPBLK	DBA1	SQR10	DF66	SQRCNT	00EF	STACK	S 0000
TSTNUM	DBAF	XFMFLG	00F1	XFORM	DE95	ZF1	DA46
ZFRO	DA44	ZTEMP1	00F5	ZTEMP3	00F9	ZTEMP4	00F7
ZXLY	DA48	_ADD1	DA98	_ADD2	DAA7	_ADDEN	DAA4
_CVB1	DC79	_CVBYT	DC76	_CVFRO	DC70	_DECIN	DCC1
_DP	DB5A	_EF1	D94F	_EF2	D96B	_EF3	D972
_EF4	D97A	_EFORM	D920	_EMIN	DBAB	_EPL	D969
_EPLUS	DBAD	_ERVAL	DA42	_EVEN	DBCE	_EXIT	DBB6
_EXIT1	DBC3	_EXPO	D91A	_F1	DD0B	_F2	DD11
_FADON	D9A9	_FAIL2	DB87	_FN1	DCB4	_FN2	DCB8
_FN3	DCA6	_FN4	D99C	_FN5	D988	_FN6	D914
_FNZER	DCA4	_FPI1	D9EA	_FRA	DD13	_FRA1	DD19
_FRADD	DA66	_FRD1	DB4E	_FRD2	DB70	_FRM	DAF1
_FRM1	DAF7	_FRM2	DB01	_FRM3	DB09	_FRS1	DB74
_FRS2	DB52	_GCHR1	DB9D	_GETCH	DB94	_GETDI	DCB9
_IFP1	D9B8	_IFP2	D9BE	_ILSHF	DA5A	_IN1	DB18
_IN2	DB1C	_INCE	DB37	_INCE2	DB3E	_IND2	DBE5
_INDON	DBE4	_MINUS	DB56	_MV1	DD36	_MV2	DD2A
_NALIC	DABE	_NDONE	DC38	_NIB1	DBED	_NIBS	DBEF
_NDINC	D94C	_NON1	DB42	_NON2	DBA3	_NONUM	DB41
_NORM	DC0A	_NOTE	DBB2	_NR	DA38	_NSH	DC10
_NSHAF	DAB5	_NXTB	DB13	_NXTB1	DC64	_NXTQ	DB43

_NORM	_COA	_NOTE	DBB2	_NR	DA38	_NSH	DC10
_NSWAP	DAB5	_NXTB	DB13	_NXTB1	DC64	_NXTG	DB43

_NXTG1	DB45	_ROUND	DA24	_RSH	DC40	_RSH1	DC48
_RSH2	DC46	_RTFAI	DBCf	_RTPAS	DBE2	_SB1	DBA5
_SBRTS	DBAC	_STCHA	DC9F	_STNUM	DC9D	_SUB	DAB3
_SUB1	DAB6	_SUB2	DAC5	_SUB3	DACE	_SWAP	DA77
_TSNFA	DBD0	_TST1	DC9C	_TSTBI	DC28	_TSTCH	DBBB
_TSTDP	DC93	_TSTN	DBDB	_TSTN1	DBD2	_TSTUN	DC31
_ZF2	DA4A	_EROV	DB26	_EXP	DB63	_EXP1	DB9B
_EXP2	DB6C	_EXP3	DB8E	_FAIL	DB65		